Asterisk

Configuración

Primeros Pasos
Poder hablar entre 2 extensiones SIP

- Una vez que asterisk está instalado, podemos configurar los archivos necesarios.
- En este caso seleccionamos como protocolo (de señalización) a SIP.
- Es necesario:
  - configurar los dispositivo SIP (en sip.conf)
  - Configurar un plan de marcación básico (en extensions.conf
- Todos estos archivos de configuración de Asterisk, se encuentran en el directorio /etc/asterisk

Primeros pasos
Configuración de dispositivo (SIP) (sip.conf)

[pedro] ;Nombre del usuario
type=friend ;Permite generar y recibir
;llamados
secret=drope ;Clave de pedro
host=dynamic ;El teléfono puede tener una ip
;dinámica
context=internos; El contexto asocia el usuario
;al plan de numeración

Primeros pasos
Plan de marcación (extensions.conf)

[internos]

exten => 101,1,Dial(SIP/pedro)

exten => 101,2,Hangup()

Primeros pasos
Lab 1

Laboratorio 1:

Interconectar 2 teléfonos utilizando el protocolo de señalización SIP

Archivos de configuración

Asterisk se configura desde múltiples archivos de configuración, cada uno para una determinada área. Los más importantes son:

- Archivo de configuración maestro: **asterisk.conf**
- Archivo de configuración de módulos: **modules.conf**
- Archivos de configuración de canales:
  - **iax.conf**: canales IAX (Inter Asterisk eXchange).
  - **sip.conf**: canales SIP.
  - **zaptel.conf/system.conf**: telefonía analógica y digital.
  - **h323.conf**: canales H323.
  - **mgcp.conf**: canales MGCP.
  - **unicall.conf**: canales R2

Configuración básica de Asterisk

- Asterisk puede ser configurado desde varios puntos. Los más importantes son:
  - desde el propio CLI
  - desde los archivos de configuración (.conf) en /etc/asterisk
- La configuración se carga al iniciar Asterisk, por lo que para aplicar cualquier cambio será necesario recargarla. Para ello basta con ejecutar el comando reload en el CLI:

CLI> reload

Archivos de configuración

- Dialplan:
  - – **extensions.conf**: el propio Dialplan.
  - – **features.conf**: dialplan para métodos. complementarios (transferencias, call parking, grabación de llamadas bajo demanda, etc.).
- Configuración de aplicaciones del dialplan:
  - – **meetme.conf**: para salas de conferencias.
  - – **musiconhold.conf**: config. de la música en espera.
  - – **queues.conf**: configuración de colas de llamadas.
  - – **voicemail.conf**: configuración de los buzones de voz.

Comunicación entre dispositivos

- **Configuración de los canales**:
  - Parámetros generales.
  - Definición de canales.
- **Configuración del plan de marcación**:
  - Definición de contextos.
  - Configuración de extensiones.

SIP.CONF: sección general

En primer lugar existe la sección [general], donde se definen variables globales y aspectos por defecto para todos los canales SIP.

- La sintaxis es la siguiente:

[general]
variable1=valor1
variable2=valor2
....
register => usuario : password @ servidorregistrar
register => ....

SIP.CONF: sección general Register

- **Register** permite a Asterisk registrar su presencia en el otro extremo. De esta forma, el proveedor sabrá la localización del cliente. En ningún caso es suficiente para poder hacer llamadas.
- El comando **Register sólo es necesario si**:
  1. Se necesita ser llamado (lo cuál implica ser localizado).
  2. Se aparece en el otro extremo con una configuración de IP dinámica.

SIP.CONF: sección general

Las variables generales más importantes son:

- **allow y disallow**: indican los codecs permitidos / no permitidos.
- **dtmfmode**: especifica el método por el cual se enviarán los tonos (dígitos pulsados durante la conversación); inband/rfc2833/info
- **nat**: informa a Asterisk el tipo de NAT en el que se encuentra; yes/no/never/route
- **externip**: dirección pública si esta atrás de un NAT.
- **context**: contexto por defecto donde entrarán las llamadas entrantes por SIP.
- **port**: puerto en el que escuchar (5060).

SIP.CONF: *type*

- **user**: envía llamadas a Asterisk.
- **peer**: recibe llamadas de Asterisk.
- **friend**: recibe y envía llamadas.

- La sintaxis para definir un *type* es:

[nombre] ; contexto
type = friend / peer / user
variable1 = valor
variable2 = valor
....

SIP.CONF: variable de canales

Las variables más importantes son:

- **type**: peer / friend
- **context**: contexto donde entrarán las llamadas generadas.
- **nat**: indica si el usuario o peer se encuentra tras un NAT.
- **host**: IP remota o dynamic (en el caso en que la IP no sea fija).
- **username**: nombre de usuario.
- **secret**: contraseña de acceso en texto plano.
- **allow y disallow**: configuraciones de codecs específicas para cada peer / friend.
- **qualify**: evalúa el estado del extremo SIP para conocer su accesibilidad y latencia (tener cuidado si el otro extremo no esta configurado para responder).
- **canreinvite**: permite que el tráfico de voz pase por el asterisk o bien directamente entre las partes.

SIP.CONF: Ej.1 Canal in/out

[pepe] ; usuario pepe
type=friend
secret=clavepepe
context=interno
host=dynamic
nat=yes

SIP.CONF: Ej.2 de Canal in/out

[juan]
type=friend
secret=clavejuan
callerid="juan perez" <1002>
context=internos
host=dynamic
nat=yes
canreinvite=no
disallow=all
allow=gsm
allow=ulaw
mailbox=1002@default

SIP.CONF: Ej.3 de Canal saliente

[sip.internetcalls.com]
type = peer
host = sip.internetcalls.com
username = miusuario
secret = miclave
dtmfmode = rfc2833
disallow = all
allow = g729
allow = g726

SIP.CONF: Ej.3 de Canal entrante

[miDID]
type = user
host = miDID.com
secret = miclave
dtmfmode = rfc2833
disallow = all
allow = g729
allow = gsm
context=interno

sip.conf

[pepe] ; usuario *type=friend ; IN/OUT* *secret=pepin ; clave* *context=internos ; rela dial plan* *callerid="pepe perez" <1001>* *host=dynamic ; IP del device* *nat=yes ; soporte NAT device* *canreinvite=no* *;disallow=all* *;allow=gsm* *;allow=ulaw* *;allow=alaw* *mailbox=1001@default*	*[juan]* *type=friend* *secret=juanin* *callerid="juan perez" <1002>* *context=internos* *host=dynamic* *nat=yes* *canreinvite=no* *;disallow=all* *;allow=gsm* *;allow=ulaw* *;allow=alaw* *mailbox=1002@default*

extensions.conf

[internos]
exten => 101,1,Dial(SIP/pepe)
exten => 101,2,Voicemail(1002,u)
exten => 101,3,Hangup()
exten => 101,102,Voicemail(1002,b)
exten => 101,103,Hangup()

exten => 102,1,Dial(SIP/juan,10)
exten => 102,2,Voicemail(1002,u)
exten => 102,3,Hangup()
exten => 102,102,Voicemail(1002,b)
exten => 102,103,Hangup()

exten => 850,1,VoiceMailMain

Ejercicio: iax.conf

[juan] *host=dynamic* *type=friend* *transfer=no* *disallow=all* *allow=gsm* *allow=ulaw* *secret=juanin* *context=internos* *qualify=yes*	*[pedro]* *host=dynamic* *type=friend* *transfer=no* *disallow=all* *allow=gsm* *allow=ulaw* *secret=clavepedro* *context=internos* *qualify=yes*

Ejercicio: voicemail.conf

[default]

1001 => 4242,Juan
1002 => 4242,Pepe

SIP.CONF: verificación con el CLI

- Mediante el comando "**reload**" en el CLI de Asterisk, se indica que recargue la configuración. Aunque es posible recargar de forma independiente sólo la conf. SIP:
  CLI> **sip reload**
- Una vez recargada, puede comprobarse los "users" que se han definido con el comando: **sip show users**
- Para ver los "peers" definidos: **sip show peers**
- Es importante notar que los "friends" son "peers" y "users" a la vez, ya que pueden recibir y enviar llamadas.
- Puede consultarse si Asterisk se ha "registrado" correctamente en los registars configurados en la sección general con el comando: **sip show registry**
- Pueden verse los canales sip activos (comunicaciones activas), vía el comando: **sip show channels**

Aplicaciones en el dialplan: IAX.CONF

- En el archivo IAX.CONF se definen todos los enlaces que se establecerán usando el protocolo IAX.
- El archivo IAX.CONF tiene la misma estructura que el SIP.CONF: una sección general y otras por canales.
- IAX tiene pequeñas diferencias que serán explicadas a continuación.

IAX.CONF: Ej. Canal in/out

[juan]
type=friend
secret=miclave
qualify=yes
port=4569
transfer=no
host=dynamic
context=interno
callerid=device <1234>

- El valor "**transfer=no**" es el equivalente a "canreinvite=no" del sip.conf, hace que el servidor Asterisk quede situado en medio de la comunicación entre clientes.

IAX.CONF: Ej. Canal in/out

[itsp] ; saliente
type=peer
qualify=yes
host=189.146.226.134

[itsp-in] ; entrante
type=user
host=189.146.226.134
context=interno

- Separa la definición del canal entrante y del saliente.

IAX.CONF: Trunking

- El protocolo IAX permite el trunking de llamadas, es decir, que múltiples streams de voz compartan un único "trunk" con otro servidor, reduciendo el overhead creado por los paquetes IP.
- Luego de 4 llamadas concurrentes comienza a haber ganancia de ancho de banda.
- Para pasar un canal IAX a modo trunk, incluir en la definición del archivo iax.conf:

  trunk=yes

EXTENSIONS.CONF

- El archivo extensions.conf es la parte central de toda la configuración, dado que es donde se define el **dialplan** de Asterisk.
- Se compone de 4 partes principales: **contextos, extensiones, prioridades** y **aplicaciones**.

EXTENSIONS.CONF: CONTEXTOS

- El dialplan se divide en secciones denominadas **contextos**, que están rotuladas y contienen un grupo de extensiones.
- Los contextos se definen colocando su nombre entre corchetes ([]). Este nombre puede contener caracteres alfanuméricos además del guión y el guión bajo. Por ej:

  [interno]
- Todas las instrucciones son parte del contexto hasta que el próximo contexto es definido.
- Existen dos contextos especiales: *[general]* (variables predefinidas) y *[globals*] (variables no predefinidas).

EXTENSIONS.CONF: EXTENSIONES

- Una extensión es una instrucción que será seguida por Asterisk, luego de ser disparada por una llamada entrante o bien por dígitos discados en un canal, definida en el marco de un contexto.
- La sintaxis de una extensión es la siguiente:

  exten => nombre,prioridad,aplicacion()
- Ejemplo:

  exten => 123,1,Answer()

EXTENSIONS.CONF: EXTENSIONES

- Extensión ***s*** (start): es una extensión especial que es utilizada si una llamada entra a un contexto sin una extensión destino específica (por ejemplo una llamada en un puerto FXO); la llamada trata de entrar automáticamente a la extensión ***s***.

  exten => s,1,Answer()

EXTENSIONS.CONF: PRIORIDADES

- Una extensión puede tener varios pasos, denominados **prioridades**.
- Las prioridades comienzan con 1 y se ejecutan en orden numérico.
- Si no existe la prioridad N + 1, Asterisk no salta a la siguiente prioridad (N+2).
- Cada prioridad ejecuta una única aplicación.
- Ejemplo:

  exten => 123,1,Answer()

  exten => 123,2,Hangup()

EXTENSIONS.CONF: PRIORIDADES

- **Prioridades sin numerar**: Asterisk introduce el uso de la **prioridad *n*** (next). Cada vez que Asterisk encuentra una prioridad n, toma el número de la prioridad anterior y le suma 1.
- Simplifica el proceso de escritura del dialplan, evitando tener que volver a numerar las prioridades al insertar una prioridad para la misma **exten**.
- Ejemplo:

```
exten => 123,1,Answer()
exten => 123,n,hago algo
exten => 123,n,Hangup()
```

Prioridades *n*

```
exten => 555,1,Answer()
exten => 555,n(LlamaPepe),Dial(SIP/pepe,20)
exten => 555,n,Voicemail(44)
exten => 555,n,Hangup

;Va a n+101 en caso de que el canal esté ocupado:
exten => 555,LlamaPepe+101,Voicemail(44,u)

;También puede tener una etiqueta esta prioridad:
exten => 555,LlamaPepe+101(PepeNoDisponible),
```

Prioridades *n*

```
exten => _044.,1,Answer
exten => _044.,n(Director),GotoIf($[${CALLERID(
  num)} = pepe]?LlamaCelular:)
exten => _044.,n,Set(TIMEOUT(absolute)=300)
exten =>
  _044.,n(LlamaCelular),Dial(Zap/1Dahdi/1/${EX
  TEN})
exten =>  _044.,n,Hangup
```

EXTENSIONS.CONF:APLICACIONES

- Las aplicaciones realizan una acción determinada en el canal actual, controlando el comportamiento de la llamada y del sistema en sí.

  Algunos ejemplos:
  - *Answer():* contesta una llamada.
  - *Hangup():* cuelga una llamada.
  - *Dial():* realiza una llamada saliente.
  - *Playback():* reproduce un archivo de sonido.
- Ciertas aplicaciones requieren del pasaje de parámetros, estos se incluyen dentro de los paréntesis, separados por “,”.

Nuestro primer dialplan: hola mundo!!

- Definir canal SIP.
- Definir pasos para llamadas entrantes:
  - Contestar la llamada.
  - Reproducir un archivo.
  - Colgar la llamada.

El primer dialplan: agregar las extensiones

- **Aplicaciones: *Answer(), Playback() y Hangup()***
- Cómo hacer un dialplan?
  - Answer() contesta un canal al que está entrando una llamada. No toma ningún argumento.
  - Playback() reproduce un archivo previamente grabado. Recibe como parámetro el nombre del archivo sin extensión, el archivo debe estar en /var/lib/asterisk/sounds
  - Hangup() cuelga el canal actual.

dialplan: extensions.conf

[interno]
exten => 100,1,Answer( )
exten => 100,2,Playback(hello-world)
exten => 100,3,Hangup( )

Más aplicaciones

- Background(). Similar a playback, pero si el usuario presiona dígitos, la aplicación lo captura y trata de enviarlo a la extensión presionada.

  exten => 123,1,Background(hello-world)

- Goto(). Nos permite mandar una llamada a otro contexto, extensión y prioridad:

  exten => 123,1,Goto(contexto,extension,prioridad)

Agregar lógica al dialplan: Background()

[interno]

exten => 101,1,Answer( ) ; contesta la llamada
exten => 101,2,Background(enter-ext-of-person) ; espera digitos y pasa a la extension en el mismo contexto
exten => 1,1,Playback(digits/1) ; reproduce el dígito 1
exten => 2,1, Playback(digits/2) ; reproduce el dígito 2

Agregar lógica al dialplan: Goto()

[interno]

exten => 4000,1,Answer( )
exten => 4000,2,Background(enter-ext-of-person)

exten => 1,1,Playback(digits/1)
exten => 1,2,Goto(4000,1) ; vuelve al inicio de la extension 4000

exten => 2,1, Playback(digits/2)
exten => 2,2,Goto(4000,1)

- Si se pasa 1 argumento a Goto(), se asume que es la prioridad dentro de la misma extensión, si se pasan 2, se asume que es otra extensión y la prioridad en el mismo contexto, y si se pasan 3, se asume que es un contexto diferente, extensión y priorida al que se quiere enviar la llamada.

Manejo de entradas inválidas y timeouts

- Cuando el usuario teclea una entrada inválida, la llamada es enviada a la extensión ***i***.
- Cuando el usuario no teclea ninguna opción, es necesario tener una extensión que nos permita manejar esta situación. Por defecto, si no se teclea nada después de 10 segundos de haberse terminado de reproducir el archivo la llamada es enviada a la extensión ***t***. Si se teclea un digito, espera 5 segundos hasta el siguiente, o numeral (#) para terminar de tipear numeros.

Agregar extensión i y t

[interno]

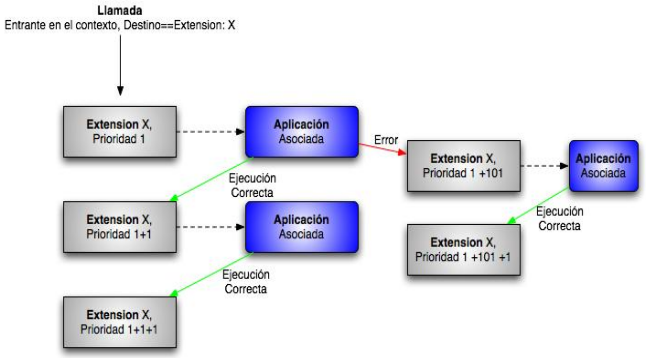
exten => 4000,1,Answer( )
exten => 4000,2,Background(enter-ext-of-person)

exten => 1,1,Playback(digits/1)
exten => 1,2,Goto(4000,1)

exten => 2,1, Playback(digits/2)
exten => 2,2,Goto(4000,1)

exten => i,1,Playback(pbx-invalid) ; llega en caso de digito invalido
exten => i,2,Goto(4000,1)
exten => t,1,Playback(vm-goodbye) ; llega en caso de no recibir digitos luego de un tiempo (10 seg)
exten => t,2,Hangup()

Flujo en el dialplan: flujo con salto de prioridades



Aplicación Dial()

- Permite conectar 2 ó más canales.
- Es especialmente útil cuando los dispositivo usan tecnologías diferentes(SIP,IAX,etc).
- Puede tomar hasta 4 parámetros:
  - El primero es el destino de la llamada en el formato *tecnologia/canal (Sip/juan, Iax2/180, Zap/1Dahdi/1)*

  exten => 4002,1,Dial(Sip/juan)
  - Se puede hacer que la llamada suene en varios canales a la vez

  exten => 4002,1,Dial(Sip/juan&Zap/1Dahdi/1&Iax/4010)

Aplicación Dial()

- El segundo argumento es el tiempo que se intentará llamar al destino. Si en ese tiempo no contestan, se pasa a la siguiente prioridad. Si no se especifica este parámetro, la llamada continúa marcando el canal, hasta que el canal es contestado o hasta que el emisor de la llamada cuelga.

  exten => 102,1,Dial(Sip/juan,10)
  - Si el usuario donde está sonando la llamada **contesta** antes de 10 segundos, **se conectan** ambos dispositivo y el dialplan termina.
  - Si el usuario **no contesta** se continúa en la **siguiente prioridad**.
  - Si el canal al que se quiere comunicar está **ocupado**, Dial() **salta a la prioridad n+101**, si es que existe.

Ejemplo de Dial

[interno]

exten => 4002,1,Dial(Sip/juan,10)
exten => 4002,n,Playback(vm-nobodyavail) ; no contesto
exten => 4002,n,Hangup()
exten => 4002,102,Playback(tt-allbusy) ; por ocupado
exten => 4002,n,Hangup()

Aplicación Dial()

- El tercer parámetro es una cadena que puede estar compuesta por varios caracteres que brindan algunas características específicas al momento de hacer la llamada. Algunos ejemplos:
  - T. Permite que quien llama pueda transferir la llamada con #
  - t. Permite que quien contesta pueda transferir la llamada con #
  - r. Genera un ring tone al que llama y espera a generar el canal de audio cuando conteste.
  - m. A diferencia de r, genera música en espera.
  - w. Permite al que llama grabar la comunicacion con *1
- En CLI, ¨core show application dial¨, muestra todas las opciones de Dial()

Llamar a los canales declarados

```
[interno]
exten => 4000,1,Answer( )
exten => 4000,2,Background(enter-ext-of-person)

exten => 4001,1,Dial(Sip/juan,10)
exten => 4001,2,Playback(vm-nobodyavail)
exten => 4001,3,Hangup()
exten => 4001,102,Playback(tt-allbusy)
exten => 4001,103,Hangup()

exten => 4002,1,Dial(Iax2/4010,10)
exten => 4002,2,Playback(vm-nobodyavail)
exten => 4002,3,Hangup()
exten => 4002,102,Playback(tt-allbusy)
exten => 4002,103,Hangup()

exten => i,1,Playback(pbx-invalid)
exten => i,2,Goto(4000,1)
exten => t,1,Playback(vm-goodbye)
exten => t,2,Hangup()
```

Aplicación Dial()

exten => 123,1,Dial(Zap/4Dahdi/4/2235980)

- Llama vía el canal número 4 de ZapDahdi al número especificado “2235980”

Variables

- En el dialplan de Asterisk existen variables, que pueden ser modificadas por el propio Asterisk en su ejecución lógica o por comandos expresos (aplicaciones) del dialplan.
- Las variables **reducen la escritura, agregan claridad al dialplan y le aportan lógica**.
- Los tipos de variables son:
  - Globales: declaradas en extensions.conf (o por comando).
  - Canal: asociadas con un canal particular.
  - Entorno: variables de entorno (UNIX Like).
- La sintaxis de una variable es:

  ${variable}

Manejo de variables

- Asignación de variables:
  - Set(Variable=valor)
  - Global(Variable=valor)
- Manejo de cadenas:
  - Subcadenas: ${Variable : offset : longitud }
    - Devuelve la subcadena de variable que comienza en offset y con la longitud especificada.
    - Ejemplos: ${123456789:2:3} devuelve 345
  - Longitud: ${LEN(Variable)}
  - Concatenación: ${Variable1}${Variable2}

Variables globales

- Permite que se pueda hacer referencia a ellas en todos los contextos, en todas las extensiones, a diferencia de las variables convencionales que sólo tienen validez en el canal actual.
- Es útil para tener claridad manejabilidad en el dialplan.
- Se pueden definir en el contexto [globals] al inicio de extensions.conf
  - [globals]
  - JUAN=Sip/juan
  - PEDRO=lax2/4010

Agregar variables al dialplan

```
[globals]
JUAN=Sip/juan          ; define la variable JUAN
PEDRO=lax2/4010

[interno]
exten => 4000,1,Answer( )
exten => 4000,2,Background(enter-ext-of-person)
exten => 4001,1,Dial(${JUAN},10)   ; hace referencia a la
   variable JUAN
exten => 4001,2,Playback(vm-nobodyavail)
exten => 4001,3,Hangup()
exten => 4001,102,Playback(tt-allbusy)
exten => 4001,103,Hangup()
```

Agregar variables al dialplan

```
exten => 4002,1,Dial(${PEDRO},10)
exten => 4002,2,Playback(vm-nobodyavail)
exten => 4002,3,Hangup()
exten => 4002,102,Playback(tt-allbusy)
exten => 4002,103,Hangup()
exten => i,1,Playback(pbx-invalid)
exten => i,2,Goto(4000,1)
exten => t,1,Playback(vm-goodbye)
exten => t,2,Hangup()
```

Variables de canal definidas automáticamente

- Listado de variables más importantes:
  - ${CALLERID}: caller ID actual, nombre y número.
  - ${CONTEXT}: contexto actual.
  - ${EXTEN}: extensión actual.
  - ${CHANNEL}: canal actual.
  - ${DIALSTATUS}: estado de la llamada: unavailable, congestion, busy, noanswer, answer, cancel, hangup.
  - ${DATETIME}: hora actual.
- Un comando útil para ver el contenido es NoOp:
  - NoOp (${VARIABLE})
    - Mostrará en el CLI el valor.

Variable ${EXTEN}

- Permite saber cuál es la extensión que fue marcada.
- Se utiliza comunmente para eliminar dígitos marcados: ${EXTEN:x}

  Si x es positivo, quita los primeros x dígitos marcados

  exten => _XXX,1,SayDigits(${EXTEN:1})

  Si x es negativo, devuelve los últimos x dígitos marcados

  exten => _XXX,1,SayDigits(${EXTEN:-1})

Patrones

- Se utilizan principalmente para llamadas salientes.
- Comienzan con el signo _
- Le dicen a Asterisk que haga match con un patrón y no con un número de extensión.
  - ***X***. Hace match con cualquier dígito de 0 a 9.
  - ***Z*** Hace match con cualquier dígito de 1 a 9.
  - ***N*** Hace match con cualquier dígito del 2 al 9.
  - ***[15-7]*** Hace match con el rango de dígitos especificados, en este ejemplo matchea los números 1,5,6,7.
  - **.** (punto) Hace match con uno o mas caracteres.
    - *exten => _NXX,1,Playback(auth-thankyou)*
- Si Asterisk encuentra más de 1 patrón para una extensión marcada, se usa la más específica:
  - *exten => _555XXXX,1,Playback(digits/1)*
  - *exten => _55512XX,1,Playback(digits/1)*

Habilitar salida de llamadas

- Se generan contextos especificos para llamadas locales a la PSTN.
- Con ello se regula y controla quiénes tienen permiso de hacer llamadas y qué tipo de llamadas pueden hacer:

```
[globals]
JUAN=Sip/juan
PEDRO=Iax2/4010
TRUNKDESALIDA=Zap/1Dahdi/1
[llamadas-locales]
exten => _9NXXXXXX,1,Dial(${TRUNKDESALIDA}/${EXTEN}:1)
exten => _9NXXXXXX,2,Congestion()
exten => _9NXXXXXX,102,Congestion()
[llamadas-larga-distancia]
exten =>
  _901XXXXXXXXXX,1,Dial(${TRUNKDESALIDA}/${EXTEN}:1)
exten => _901XXXXXXXXXX,2,Congestion()
exten => _901XXXXXXXXXX,102,Congestion()
```

Include

- Puede utilizarse un contexto dentro de otro contexto a través de la directiva ***include***
- Permite habilitar derechos de acceso a las diferentes secciones del dialplan.
- Por ejemplo, que los dispositivo del contexto [internos] puedan hacer llamadas por la red PSTN

  include => context

- Primero trata de encontrar las extensiones en el contexto actual.
- Si no la encuentra, trata de encontrarla en el primer contexto incluido, y después en el segundo y así sucesivamente.

Agregar include al dialplan

[internos]

include => llamadas-locales

include => llamadas-larga-distancia

Manipulación de expresiones y variables

- Las expresiones son una combinación de variables, operadores y valores que arrojan un resultado.
  Sintaxis:

  $[expr1 operador expr2]
- Operadores lógicos: |(or) , &(AND)
- Operadores de comparación: =, !=, <, >, <=, >=
- Operadores aritméticos: +, -, *, /, %
- [...] , Ejemplo:

  exten => 4003,1,Set(COUNT=3)

  exten => 4003,2,Set(NEWCOUNT=$[${COUNT} + 1])

  exten => 4003,3,SayNumber(${NEWCOUNT})

Bifurcación condicional

- Permite tomar desiciones dentro del dialplan.
- Aplicación GotoIf()
  - *GotoIf(expresion1?destino1:destino2)*
- Si la expresión evaluada es verdadera, la llamada es enviada a destino1, de lo contrario es enviada a destino2.
- Una cadena vacía y el número 0 son evaluados con falso, cualquier otro valor es verdadero.
- Cualquiera de los destinos puede ser omitido, pero debe estar alguno de los 2.
- Si el destino omitido es el camino que debe seguir la llamada, el flujo que se sigue es la siguiente prioridad dentro de la extensión actual.

Ejemplo de GotoIf()

```
exten => 104,1,Set(TEST=1)
exten => 104,2,GotoIf($[${TEST} = 1]?10,20)
exten => 104,10,Playback(weasels-eaten-phonesys)
exten => 104,20,Playback(office-iguanas)

exten => 105,1,Set(COUNT=10)
exten => 105,2,GotoIf($[${COUNT} > 0 ]?:10)
exten => 105,3,SayNumber(${COUNT})
exten => 105,4,Set(COUNT=${COUNT} - 1)
exten => 105,5,Goto(2)
exten => 105,10,Hangup()

exten => 106,1,GotoIf($[${CALLERIDNUM} = 8842374]?20:10)
exten => 106,10,Dial(Sip/Juan)
exten => 106,20,Playback(abandon-all-hope)
exten => 106,21,Hangup()
```

Bifurcación condicional basada en tiempo

- Verifica la hora actual del servidor, permitiendo tomar decisiones basadas en tiempo.
- Se utiliza cuando se quiere dar una bienvenida diferente en horarios de trabajo y fuera de trabajo.

GotoIfTime(hora,dias_de_semana,dias_del_mes,meses?etiqueta)

- Envía la llamada a etiqueta si la fecha y hora actual concuerdan con el criterio especificado por los parámetros.

Bifurcación condicional basada en tiempo

- hora. Lista de uno o más rangos de horario en formato de 24 horas.
  09:00-17:00
- dias_de_semana. Lista de uno o más días de la semana
  mon, tue
- dias_del_mes. Día numérico del mes
  7-12,15
- meses. Lista de uno o más meses del año
  jun, apr, jul
- * Matchea con cualquier valor
- Etiqueta puede ser una prioridad dentro de una misma extensión, una prioridad y extensión dentro del mismo contexto o un contexto, extensión y prioridad.
- Exten => s, 1, GotoIfTime(*,*,2, nov?open,s,1) ; se envía al contexto open, extensión s, prioridad 1

Correo de voz

- Se permite un número ilimitado de buzones.
- Notificación por correo:
  - Puede anexar el mensaje de voz (.wav)
  - Indicador luminoso o señal de mensaje en espera.
  - Indicador auditivo de mensaje en espera al levantar el teléfono.
  - Manejo de mensajes vía telefónica.
- Los contextos de voicemail son definidos de igual manera que en dialplan.

Correo de Voz

- mailbox => password, nombre [,e-mail [, pager-email [, opciones]]]
- mailbox: número de mailbox, generalmente asociado al número de extensión.
- password: clave numérica del buzón.
- nombre: nombre del propietario del buzón. Se utiliza para permitir incorporar en el directorio de Asterisk.
- e-mail: correo electrónico adonde se envía la notificación.
- pager-email: correo electrónico para enviar la notificación.
- Opciones: hay varias como attach=yes
- Voicemail() envía al emisor al buzón especificado para dejar un mensaje.
- El número de buzón puede ser proseguido por la letra ***b*** o ***u.*** Con la letra ***b,*** se le informa al emisor que el usuario está ocupado. Con ***u*** que no esta disponible.

voicemail

- **Voicemail.conf**
  - [default]
  - 4001 => 1234,Juan Perez,juan@dominio.com,
- **Extensions.conf**
  - exten => 4001,1,Dial(${JUAN},10) ; hace referencia a la variable JUAN
  - **exten => 4001, 2,voicemail(4001@default,u)**
  - **exten => 4001,102,voicemail(4001@default,b)**
- **Acceso al buzón de voz**
  - **Exten => *98, 1, voicemailMain()**

Sala de conferencias

- Permite crear conferencias protegidas por clave.
- Administrar conferencias.
- Callar o expulsar a un miembro de la conferencia.
- Crear conferencias estáticas (en el meetme.conf).
- Crear conferencias dinámicas (utilizando el plan de marcación).
- **Meetme.conf**
  - [rooms]
  - Conf => 600
- **extensions.conf**
  - exten => 600,1,meetme(600, i, 54321)
  - Se anuncia cuando alguien entra o sale de la conferencia.
  - exten => 601,1,playBack(conf-thereare)
  - exten => 601,2,meetmeCount(600)
  - exten => 601,3, playBack(conf-peopleinconf)

Interconexión entre Asterisk IAX

- Una conexión IAX entre dos Asterisk se establece en los siguientes pasos:
  - Configurar en ambos servidores el archivo iax.conf, uno como peer y otro como user.
  - - Modificar el dialplan del user para que se puedan efectuar llamadas desde el user al peer.
  - - En el caso que la IP sea dinámica, se debe registrar el peer con el user.
  - - Repetir los pasos anteriores en la dirección opuesta (intercambiar peer y user), para que ambos pueden enviar y recibir llamadas.
- En el próximo ejemplo se conectarán dos servidores Asterisk A y B, de manera tal que ambos puedan llamarse entre sí, utilizando autenticación con clave simétrica.

IAX – Autenticación con clave simétrica

Configuración para el servidor A:

iax.conf

```
[serverB_in]
 type=user ; llamadas del serv B al *
 auth=md5
 secret=passwordA
 context=from-iax ; conexto que entran las llamadas de B

[serverB_out]
 type=peer
 host=serverB.domain.com
 auth=md5
 secret=passwordB
 username=serverA_in
```

IAX – Autenticación con clave simétrica

- La sección [serverB_in] permite al servidor A recibir llamadas por parte del servidor B. El user indicado por el servidor B debe hacer match con el nombre de esta sección. En secret se indica el password para realizar la autenticación y en context el contexto al cuál serán derivadas las llamadas entrantes del servidor B vía este canal.
- La sección [serverB_out] permite al servidor A realizar llamadas al servidor B. En host se debe indicar la dirección IP o nombre DNS del servidor B, o bien, la palabra “dynamic” si la IP es dinámica. En este útimo caso, el peer debe realizar un comando register para poder ser localizado por el cliente.
- **extensions.conf**
  - exten => _7XXX,1,Dial(IAX2/serverB_out/${EXTEN:1},30)
  - exten => _7XXX,2,Congestion
- Esta configuración permite que al discar 7XXX en el servidor A, se haga una llamada al servidor B con ese número pero quitándole el prefijo 7.

IAX – Autenticación con clave simétrica

- Configuracion para el servidor B:
- **iax.conf**

```
[serverA_in]
 type=user
 auth=md5
 secret=passwordB
 context=from-iax
[serverA_out]
 type=peer
 host=serverA.domain.com
 auth=md5
 secret=passwordA
 username=serverB_in
```

- **extensions.conf**

```
exten => _8XXX,1,Dial(IAX2/serverA_out/${EXTEN:1},30)
exten => _8XXX,2,Congestion
```

IAX – Autenticación con RSA

- El protocolo IAX2 soporta autenticación fuerte con claves de encripción asimétricas, utilizando RSA. Esto permite tanto la autenticación de un usuario con Asterisk como también la de Asterisk con algún proveedor.
- La generación de claves RSA es realizada con la utilidad **astgenkey** provista por Asterisk. Para crearlas simplemente ejecutar en la línea de comandos:
  - # astgenkey –n nombre_clave
- Se generarán dos archivos nombre_clave.pub y nombre_clave.key en el directorio actual. Estas claves deben copiarse a /var/lib/asterisk/keys, dado que allí Asterisk busca las claves para realizar la autenticación.

IAX – Autenticación con RSA: Ej

- En este caso, el proveedor (peer) sólo recibe llamadas. En el archivo iax.conf del mismo se debe colocar la siguiente información del usuario para permitir la autenticación del cliente:

[usuario_in]
type=user
auth=rsa
inkeys=clave_usuario
context=from-iax

- En el valor inkeys, se debe poner el nombre de la clave pública del usuario. Esta clave debe copiarse previamente al proveedor.

IAX – Autenticación con RSA

- Asimismo, en este caso, el cliente (user) sólo realiza llamadas. En el archivo iax.conf del mismo se debe colocar la siguiente información del usuario para permitir la autenticación con el proveedor:

[proveedor_out]
type=peer
host=proveedor.dominio.com
auth=rsa
outkey=clave_usuario
username=usuario_in

- En el valor outkey, se debe poner el nombre del set de claves del usuario.

IAX – Autenticación con RSA

- Los siguientes comandos del CLI están relacionados con el manejo de claves:
  - **init keys**: inicializa las claves RSA y pregunta por las passphrases si es necesario.
  - **show keys**: muestra información sobre las claves RSA instaladas.

Curso elaborado por
Júlian Dunayevich, Lázaro Baca, Andrés Brassara y Santiago Alberch
julian@dunayevich.com
lazaro.baca@gmail.com
abrassara@gmail.com
salberch@gmail.com

Basándose en:

Irontec: **contacto@irontec.com** (CC)
Asterisk, The Future of Telephony, Jim Meggelen, Jared Smith, and Leif Madsen, O´REILLY, 2005